Adaptive Framework for Automated Mapping and Architecture Trades for Embedded Heterogeneous Systems

Raju D Venkataramana
Tandel Systems, LLC

This paper presents a framework that can be adapted to perform automated mapping/scheduling and architecture trades for embedded computing environments. The framework is based on a learning automata model whose adaptable nature makes it invaluable in tools that provide end-to-end solutions for embedded systems development. The framework has been incorporated into one such tool, Systems and Applications Genesis Environment (SAGE). The paper begins with an introduction to learning automata and the architecture of the framework. The following sections discuss how algorithms for automated mapping/scheduling and architecture trades are constructed within the framework.

2.0 Learning-Automata based Framework

The framework is based on a variable structure stochastic automaton (VSSA). The VSSA guarantees robust behavior in the absence of complete knowledge of the solution space, and has rigorous mathematical properties that can be exploited to develop efficient algorithms. It can be viewed as a stochastic finite state machine with a set of actions and associated probabilities that help learn the nature of an unknown environment. For every random action that is chosen, the environment that needs to be learned provides a response that is stochastically related to the chosen action. The iterative process of choosing random actions and recording the responses is continued until the solution space is satisfactorily explored, as depicted in Figure 1. A learning automaton is usually represented as a quintuple $\{\Phi,\alpha,\beta,A,G\}$, it's mathematical representation is defined below. The VSSA however can be mathematically simplified such that each state corresponds to a distinct action and hence the output function mapping G becomes an identity mapping. Hence, the VSSA becomes a triple $\{\alpha,\beta,A\}$. The nature of the response "β", determines if the VSSA is a P-model, Q-model or S-model automaton [1]. The learning algorithm A, drives the reinforcement scheme that controls the probability vector of the actions.

Mathematical representation:

- The state of the automaton at any iteration 'n', denoted by $\phi(n)$ is an element of the finite set
  $\Phi = \{\phi_1,\phi_2,...,\phi_s\}$
- The output of the automaton at the iteration 'n', denoted by $\alpha(n)$, is an element of the set
  $\underline{\alpha} = \{\alpha_1,\alpha_2,...,\alpha_r\}$
- The input to the automaton at the iteration 'n', denoted by $\beta(n)$, is an element of the set
  $\underline{\beta} = \{\beta_1,\beta_2,...,\beta_m\}$
  Or $\underline{\beta} = \{(a, b)\}$
- A, is the updating algorithm or the reinforcement scheme.
- The output function G(.), determines the output of the automaton at any iteration 'n' in terms of the state at that iteration.
  $\alpha(n) = G[\phi(n)]$
- Reinforcement Scheme:
  $P(n+1) = A(P(n), \alpha(n), \beta(n))$
  Where P(n) is the probability vector at iteration 'n'.

Our framework consists of a set of independent learning automata. Each with their own set of actions but with a common learning algorithm. The response from the environment is translated to individual responses for each of the automata based on different heuristics. The framework adapted to the automated mapping algorithm is depicted in Figure 2. The important elements in the framework are the environment, actions for each of the automata, and the reinforcement scheme or learning algorithm.

3.0 Automated Mapping/Scheduling

In automated mapping/scheduling, the objective is to match and schedule application tasks to a heterogeneous suite of machines such that pre-defined performance criteria such as size, weight, power, latency and bandwidth are optimally satisfied. The learning automata framework can be adapted to achieve this objective. A pictorial representation of this schema is shown in Figure 2.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
21 MAY 2003	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Learning Automata	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Tandel Systems, LLC**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**14**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Environment: Consider a Heterogeneous Computing (HC) system model. It consists of the application represented as a task flow graph (TFG) and the target architecture represented as a processor graph (PG). Different cost metrics can be defined for the HC system model [2].
Automata Construction: Every task in the TFG is associated with an automaton. The set of actions for each of the automata is the set of processors (nodes) in the PG. An action therefore corresponds to mapping a task to a processor. The probability equations for the reinforcement scheme and five heuristics which were developed to drive the learning algorithm are omitted here due to space restrictions and can be found in [2]. Results of the mapping algorithm are shown in figures 4 and 5. The performance metric in these graphs is "Minimizing the total execution time". The behavior of the algorithm is plotted for varying weights and number of application tasks. In Figure 4, the communication complexity is assumed to be medium. It can be seen that as weight is reduced and the number of tasks increased, the optimality in the chosen performance metric is lost. The graph also depicts the graceful degradation in performance. In Figure 5 the communication complexity is assumed to be low and similar observations can be made.
Key Benefits: The automated mapping algorithm within the proposed framework provides several benefits. The primary advantage is the ability to define multiple cost metrics that drive the optimality of the system individually. The construction of the algorithm is such that the system model is made independent of the learning automata model, making it universally applicable to any application domain. This feature allows the mapping scheme to be incorporated as a plug-in into a tool for application development for embedded systems like SAGE, thereby adding value to the tool.

4.0 Architecture Trades

An architecture trade requires the automatic construction of a system that minimally conforms to the design specifications provided by the user. This architecture can then be fine-tuned to match the exact system requirements. The proposed framework can be used to develop algorithms that aid the architecture trades process. Figure 3 depicts the adaptation of the framework to this trades process.
Environment: The environment here is the detailed architecture for an embedded computing system. The system is analyzed for design requirements and a figure of merit is used to reflect the health of the architecture. The figure of merit is used to generate a response that is fed to the automata model.
Automata Construction: Here the user typically has to specify the component types that define his target system. An automaton is then associated with every component type contained in the target system. For instance, if the system architecture consists of a general purpose processor, two ASIC chips, a memory module and a network bus, then the automata model will contain an automaton each for the processor, each of the ASIC chips, the memory module and the network bus. The actions for the automata will correspond to the different classes of components that are available under each type. For example, if the available processor classes are Pentium III, FPGA, and PowerPC 603, then the automaton for the processor will have three actions corresponding to each of the processors.

When each automaton selects an action at random, the resultant system forms the target for the application. The target is analyzed for design requirements and the response is used to drive the algorithm iteratively until an optimal solution is reached.
Key Benefits: As can be seen from the construction, the learning automata framework that was used for automated mapping/scheduling is easily adapted for architecture trades. This particular feature of the framework makes it invaluable for tools that provide end-to-end solutions. Due to its randomized nature, the proposed methodology allows the designer to better explore the design space.

5.0 Conclusion

In conclusion, we propose a framework that can be adapted to perform automated mapping/scheduling and architecture trades for heterogeneous systems. The framework is based on an automata model that provides critical advantages over existing systems for both mapping and trades study. The utility of this framework demands special attention in the context of incorporating into application development tools that provide complete end-to-end solutions. This capability has been incorporated into SAGE [3], a systems integration framework for embedded systems.

References:

1. K. Narendra and M.A.L. Thathachar, *Learning Automata: An Introduction*, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

2. R.D. Venkataramana and N. Ranganathan, “Multiple Cost Optimization for Heterogeneous Computing Systems Using Learning Automata”, Proceedings of the HCW, pp 137-145, April 1999.
3. M. Patel and K.L Jordan, “SAGE: An Application Development Tool Suite for High Performance Computing Systems”, 2000 IEEE Aerospace Conference, Mar 18-25, Montana.

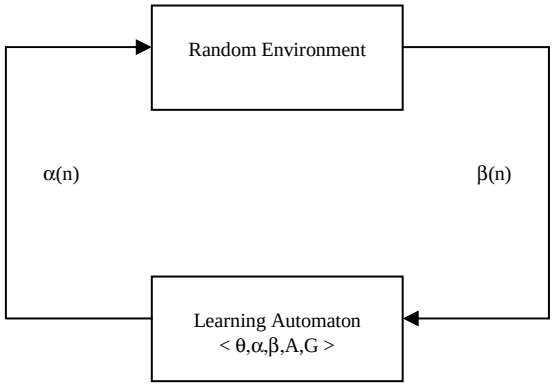


Figure 1: Learning Automaton Model

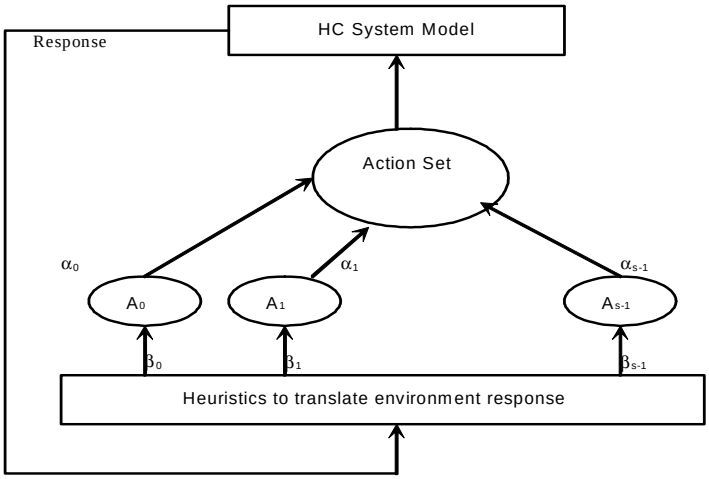


Figure 2: Framework for Automated Mapping/Scheduling

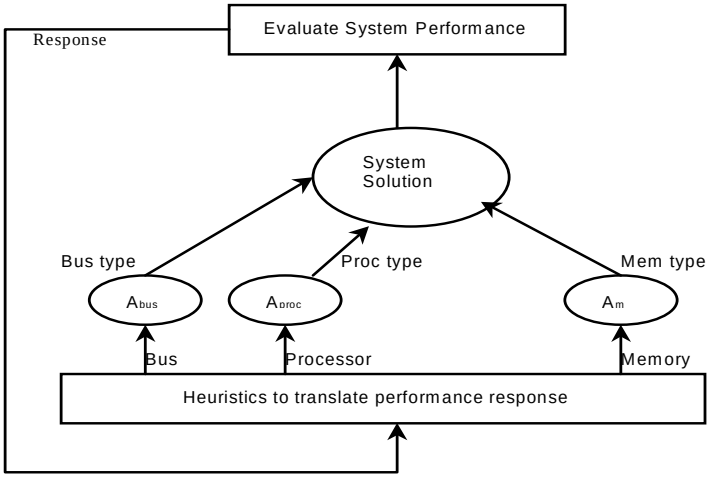


Figure 3: Framework for Architecture Trades

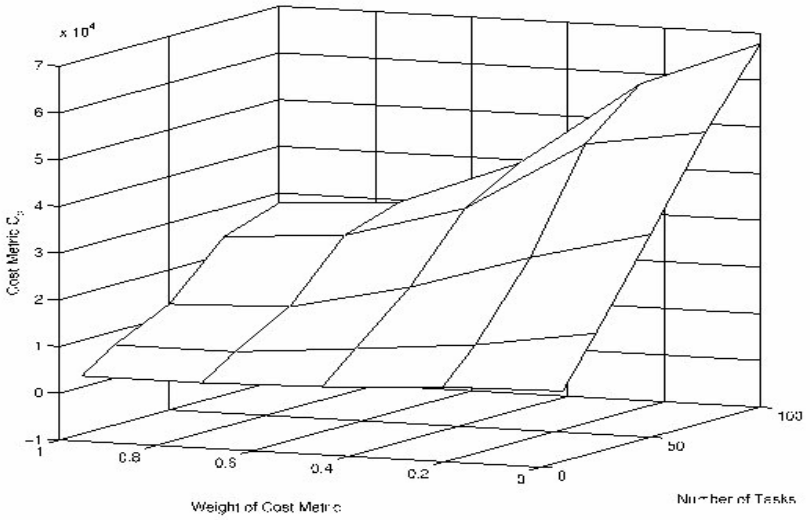
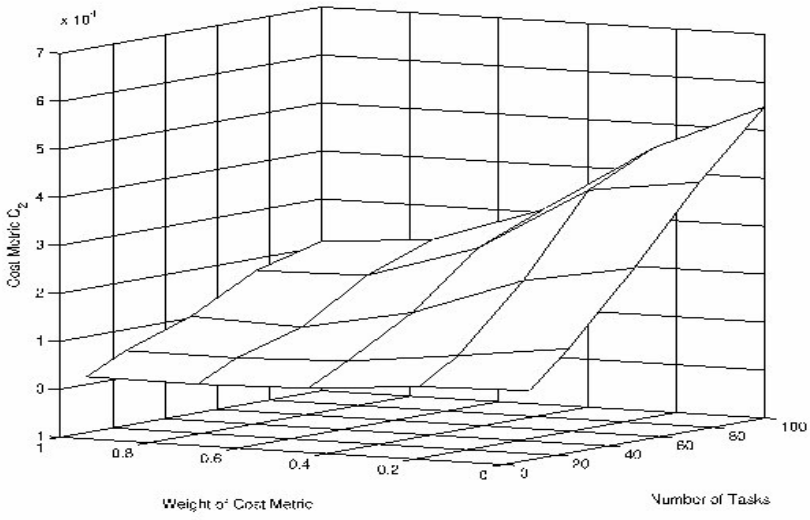


Figure 4: Performance plot of mapping algorithm for medium communication complexity



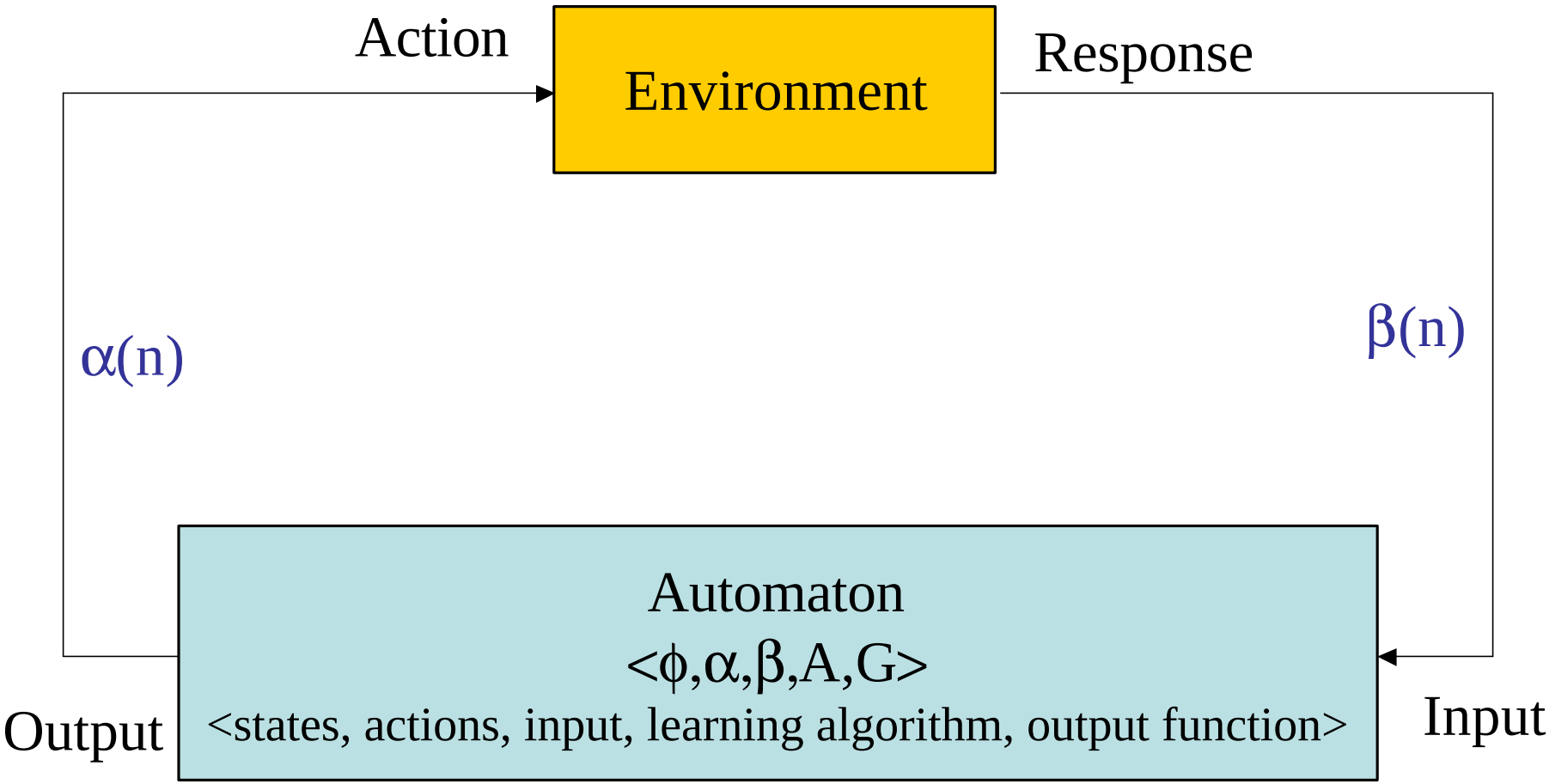
Figre 5: Performance plot of mapping algorithm for low communication complexity

Learning Automata

- Learns the unknown nature of an environment
- Variable structure stochastic learning automaton is a quintuple $\{\varphi,\alpha,\beta,A,G\}$ where:
  - $\varphi(n)$, state of automaton; $\varphi=\{\varphi_1,\ldots,\varphi_s\}$
  - $\alpha(n)$, output of automaton; $\alpha=\{\alpha_1,\ldots,\alpha_r\}$
  - $\beta(n)$, input to automaton; $\beta=\{\beta_1,\ldots,\beta_m\}$
  - A, is the learning algorithm;
  - $G[.]$, is the output function; $\alpha(n)=G[\varphi(n)]$

  n indicates the iteration number.

Learning Automaton Schematic



Probability Vector

- $p_j(n)$, action probability; the probability that automaton is in state j at iteration n.
- Reinforcement scheme

  If $\alpha(n) = \alpha_i$ and for j <> i; (j=1 to r)

  $p_j(n+1) = p_j(n) - g\,[p_j(n)]$ when $\beta(n)= 0$.

  $p_j(n+1) = p_j(n) + h\,[p_j(n)]$ when $\beta(n)= 1$.
- In order to preserve probability measure,

  $\Sigma\ p_j\,(n) = 1$, for j = 1 to r.

contd ...

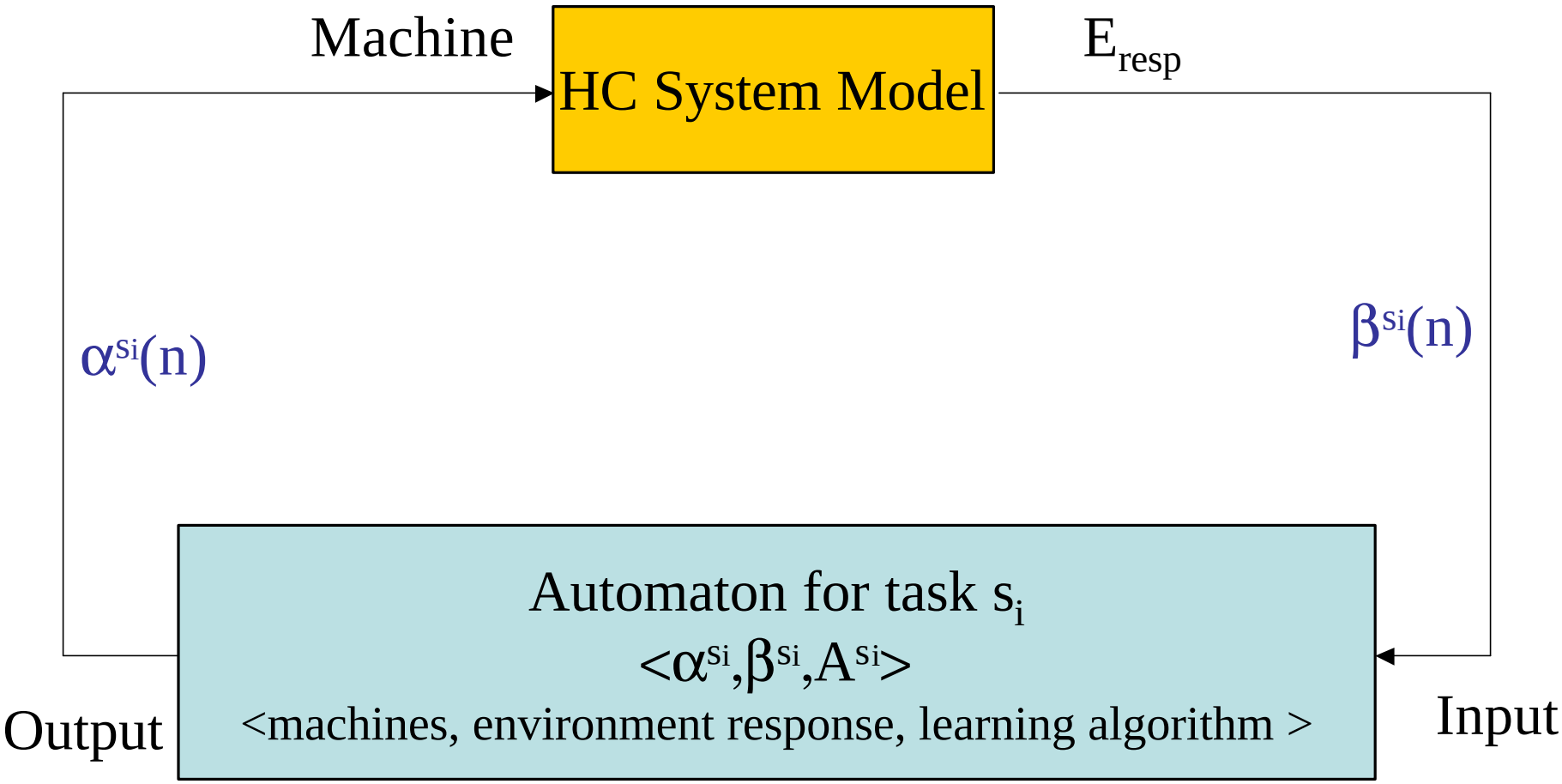
- If $\alpha(n) = \alpha_i$

$$p_i(n+1) = p_i(n) + \sum_{j=1,\, j<>i}^{r} g(p_j(n)) \quad \text{when } \beta(n) = 0$$

$$p_i(n+1) = p_i(n) - \sum_{j=1,\, j<>i}^{r} h(p_j(n)) \quad \text{when } \beta(n) = 1$$

- $g(.)$ is the reward function
- $h(.)$ is the penalty function

Schematic of Proposed Automata Model for Mapping/Scheduling



Model Construction

- Every task s_i associated with an S-model automaton (VSSA).
- VSSA represented as $\{\alpha^{si}, \beta^{si}, A^{si}\}$, since r = s
  - α^{si} is set of actions $\alpha^{si} = m_0, m_1, \ldots, m_{|M|-1}$
  - β^{si} is input to the automaton, $\beta^{si} \in [0, 1]$

    closer to 0 – action favorable to system;

    closer to 1 – action unfavorable to system
  - A^{si} is reinforcement scheme
- $p_{ij}(n)$ - action probability vector
  - probability of assigning task s_i to machine m_j

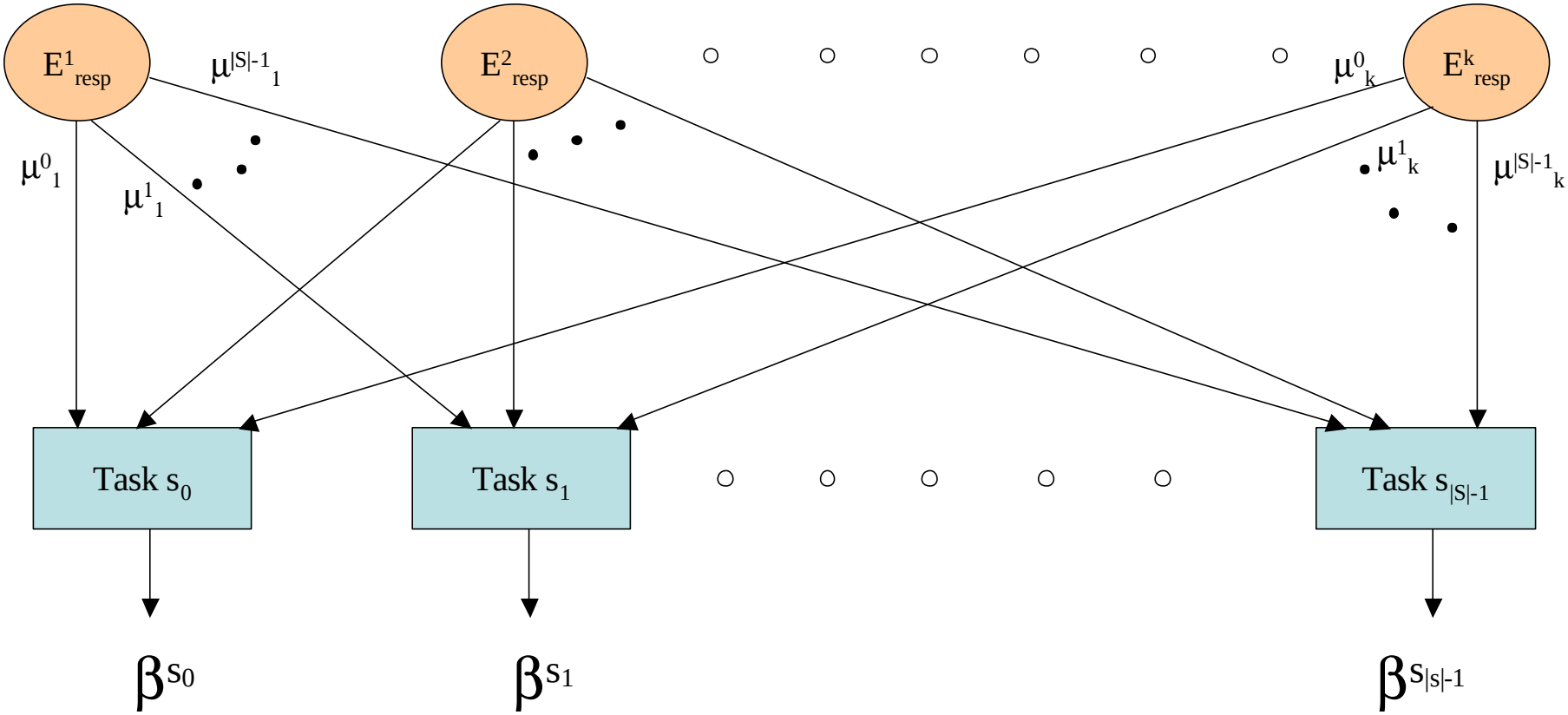
contd …

- Automata model for Mapping/Scheduling
  - S-model VSSA is used
  - Each automaton is represented as a tuple $\{\alpha^{si}, \beta^{si}, A^{si}\}$
  - $\alpha^{si} = m_0, m_1, \ldots, m_{|M|-1}$
  - $\beta^{si} \in [0, 1]$

    (closer to 0 - favorable, 1 - unfavorable)
  - If $c_k(n)$ is better than $c_k(n-1)$

    $E^k_{resp} = 0$ else $E^k_{resp} = 1$
  - Translating E^k_{resp} to $\beta^{si}(n)$ requires two steps

Translating E^{k}_{resp} to β^{s_i}



contd …

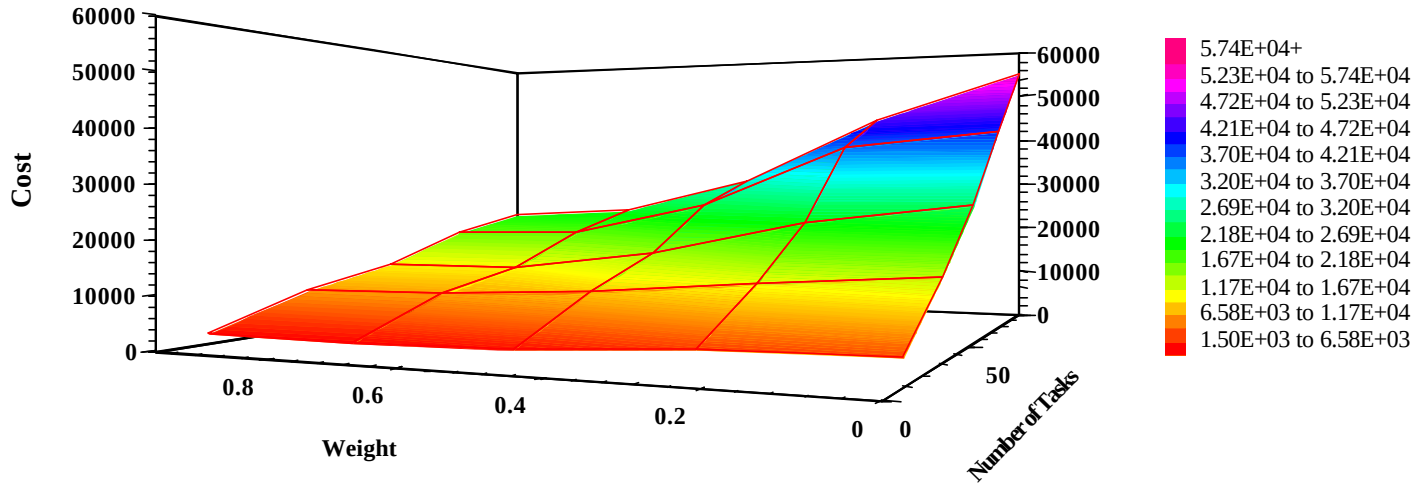
- Step 1: Translate E^k_{resp} to $\mu^{si}_k(n)$, where
  - $\mu^{si}_k(n)$ - input to automaton s_i with respect to cost metric c_k
  - achieved by the heuristics

- Step 2: Achieved be means of Lagrange's multiplier

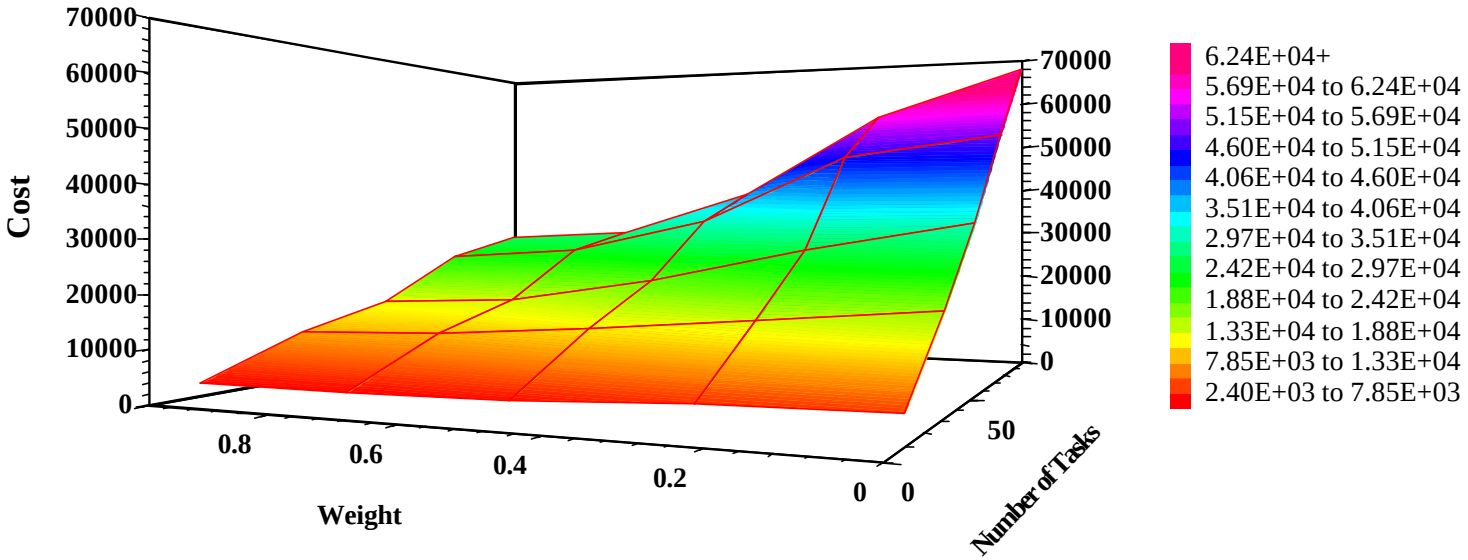
$$\beta^{si}(n) = \sum_{j=1}^{|C|} \lambda_k * \mu^{si}_k(n), \; i=1 \text{ to } |S|-1; \quad \sum_{j=1}^{|C|} \lambda_k = 1, \; \lambda_k > 0$$

where λ_k is the weight of metric c_k

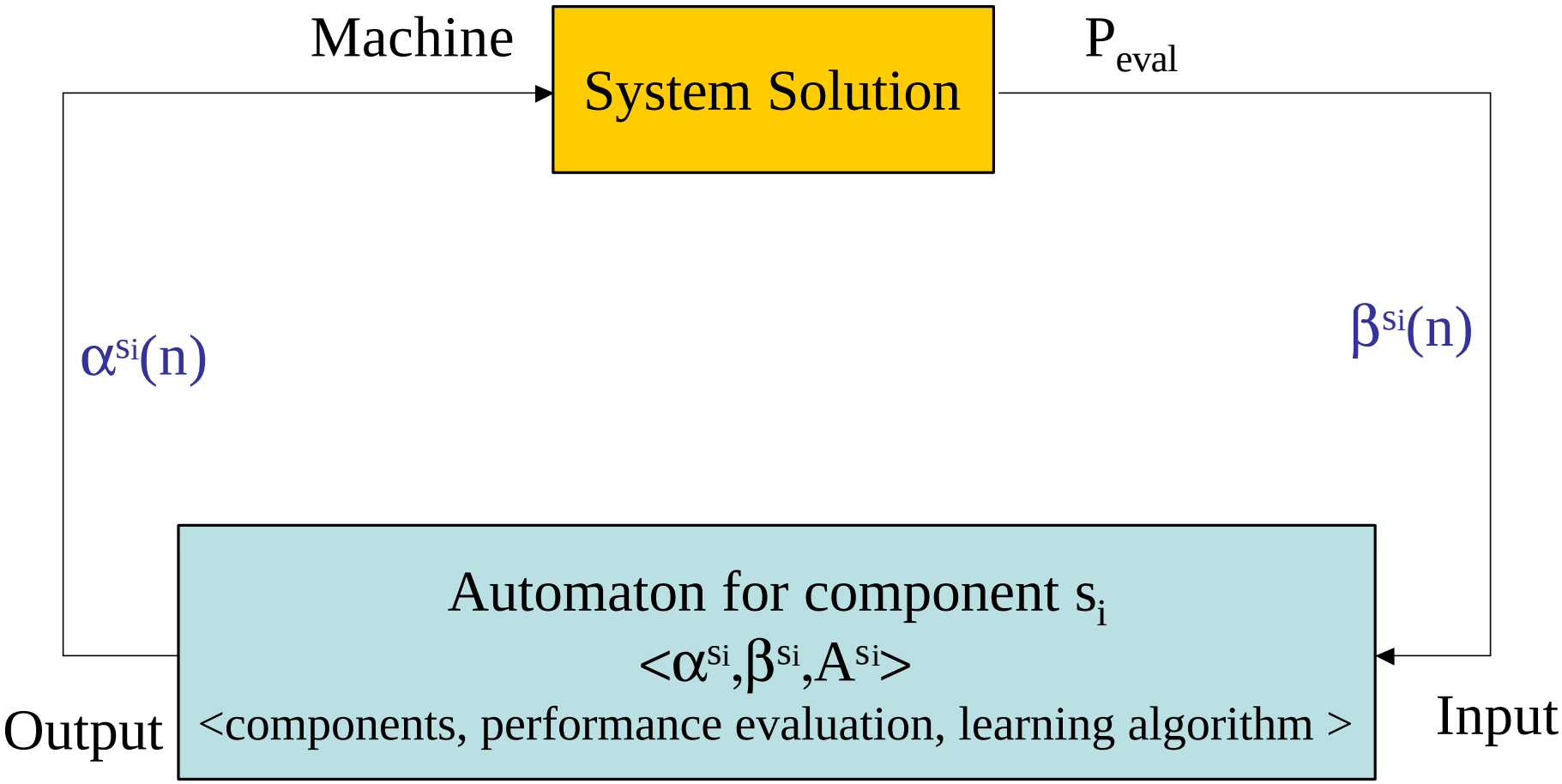
Low Communication Complexity, Machines = 5



Medium Communication Complexity, Machines = 5



Schematic of Proposed Automata Model for Architecture Trades



Model Construction

- Every component of the HW system s_i associated with a P-model automaton (VSSA).
- VSSA represented as $\{\alpha^{si}, \beta^{si}, A^{si}\}$, since r = s
  - α^{si} is set of component types $\alpha^{si} = c_0, c_1, \ldots, c_{|M|-1}$
  - β^{si} is input to the automaton, $\beta^{si} = 0, 1$

    0 – performance favorable to system; 1 – unfavorable to system
  - A^{si} is reinforcement scheme
- $p_{ij}(n)$ - action probability vector
  - probability of choosing component s_i from component type c_j

contd …

- Automata model for Architecture Trades
  - P-model VSSA is used
  - Each automaton is represented as a tuple $\{\alpha^{si}, \beta^{si}, A^{si}\}$
  - $\alpha^{si} = c_0, c_1, \ldots, c_{|M|-1}$
  - $\beta^{si} \in 0, 1$

    (0 - favorable, 1 - unfavorable)
  - If $c_k(n)$ is better than $c_k(n-1)$

    $P_{eval} = 0$ else $P_{eval} = 1$

Conclusions

- Adaptive Framework for Mapping and Architecture trades
- Automata models allow optimization of multiple criteria
- Efficient / gracefully degradable solutions
- Framework construction suitable for tool integration
  - Mapping algorithm integrated with **SAGE™**
- Provides a basis for systems design from application to the embedded HW